Chapter 35: Free Simulation Software and Library

Barkan Ugurlu and Serena Ivaldi

Abstract With the advent of powerful computation technologies and efficient algorithms, simulators became an important tool in most engineering areas. The field of humanoid robotics is no exception; there have been numerous simulation tools developed over the last two decades to foster research and development activities. With this in mind, this chapter is written to introduce and discuss the current-day open source simulators that are actively used in the field. Using a developer-based feedback, we provide an outline regarding the specific features and capabilities of the open-source simulators, with a special emphasis on how they correspond to recent research trends in humanoid robotics. The discussion is centered around the contemporary requirements in humanoid simulation technologies with regards to future of the field.

Key words: dynamic simulation, physics engine, open source software, humanoid modeling.

1 Introduction

Owing to their anthropomorphic structure, humanoid robots have the potential to replace humans in a variety of different tasks. The main motivation in pursuing human-like robots is that numerous man-made tasks are in fact quite hazardous and risky for human life; therefore, a robotic counterpart might reduce these undesired

Barkan Ugurlu
Dept. of Mechanical Engineering, Ozyegin University, 34794 Istanbul, Turkey
e-mail: barkan.ugurlu@ozyegin.edu.tr

Serena Ivaldi
Inria, Villers-les-Nancy, F-54600, France
& Intelligent Autonomous Systems Lab, TU Darmstadt, Germany
e-mail: serena.ivaldi@inria.fr

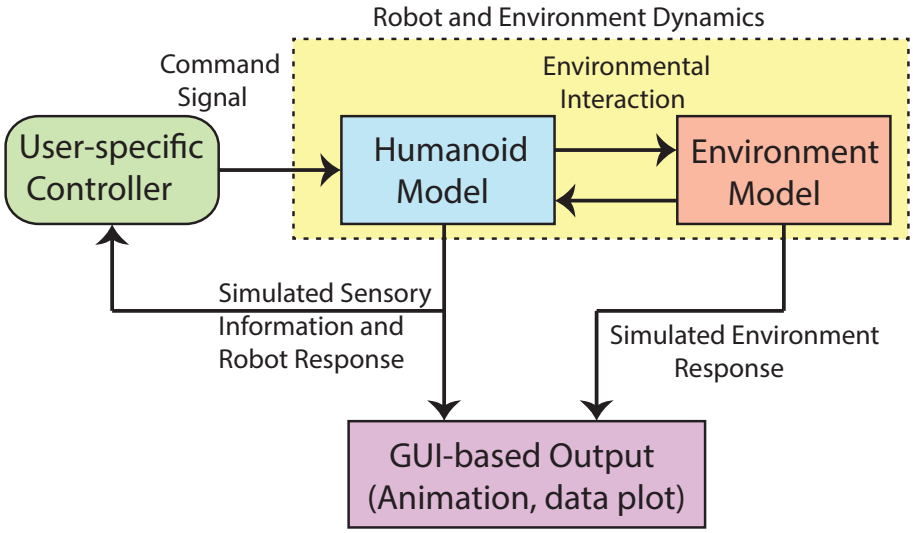

Fig. 1 Fundamental elements of a humanoid robot simulator.

factors to a technologically possible minimum. With this goal in mind, research activities on humanoid robotics have sparked in the last two decades.

In order to foster activities in this popular research area, real-time simulators have been actively used for numerous reasons. Fast and flexible prototyping, optimized actuator selection, robotics education, fail-safe validation of controllers and trajectory planners can be listed as some of the distinctive advantages, gained through simulation software. One can simply generate a virtual world in which a simulated humanoid may realize a series of specific tasks that can be difficult to prototype, test and realize directly in the physical world. Simulators allow us to complete the preliminary part of the research before initiating real-world experiments.

Fig. 1 illustrates the fundamental elements regarding a humanoid robot simulation software. The most basic elements are the humanoid model and the environment model. The humanoid model contains the information that is specific to the simulated humanoid in question, such as, number of DoFs (Degrees of Freedom), mechanical parameters, sensors, geometric structure, actuator properties, etc. The environment model consists of the physical properties of the virtual world. Namely, they are gravitational acceleration, terrain type and surface structure, environmental structures (e.g., walls, furniture, tools), extra objects in addition to the simulated humanoid, ground stiffness and friction.

The simulator should be able to compute all the resultant physical quantities that occur due to dynamic interactions between the simulated robot and the virtual environment for a given period of time. Its degree of realism strongly depends on how the developer accurately characterized the robot-environment dynamics as it is run to obtain the result of simulation.

Furthermore, a user-specific controller sends command signals to the simulated humanoid, to make it perform the desired motion pattern. It may also evaluate the simulated robot response by means of real-time feedback.

Simulated humanoid and environment responses are usually output via a GUI screen that may include an animation window to observe robot behavior and task-specific plots to monitor variations of important parameters.

When looking at Fig. 1, one may realize that the simulation design has two intrinsic challenges: i) development of computationally fast and efficient numeric solvers, ii) precise and accurate dynamics algorithms concerning the simulated robot and environment.

Until the mid 1990s, researchers attempted to overcome the first challenge by implementing alternative computation techniques like parallel algorithms [1–6]. A recent benchmark study can be found in [7]. Nevertheless, the fast and ever-growing pace of semiconductor technology paved the way for highly powerful computation tools, and therefore, seeking alternative computation techniques has not been a major issue.

While advances in computer technology enabled humanoid roboticists to overcome the first challenge, it is their cumulative research effort that is responsible for the formulation of precise and accurate dynamics algorithms. A milestone study was conducted by Luh et al. to address an online computation scheme for a robotic manipulator [8]. To mention some other major contributions, Featherstone proposed the articulated body inertia method [9]. Khatib proposed operational space formulation to map joint space dynamics into task space dynamics [10]. Rodriguez made use of Kalman filters to implement powerful dynamics algorithms [11]. On top of these prior works, various efficient dynamics algorithms have been developed so far, for instance, see [12] as an exemplary recent work.

To the best of authors' knowledge, Walker and Orin composed all the necessary elements in a unifying framework and developed one of the first robotic simulators [13]. Researchers followed Walker and Orin's footsteps and derived floating-base dynamics [14–16], so as to provide real-time humanoid simulation software, e.g., see [17–19].

Dynamic simulation has become a powerful tool for prototyping complex behaviors and study new robot control algorithms, especially for humanoid robots. The robotics teams competing in the DARPA Robotics Challenge made extensive use of such tools, from simulating contacts and whole-body movements to planning complex manipulations. Many different simulation tools have been developed to meet the needs of the researchers. More generally, the number of tools for dynamics simulation has grown substantially in the last years. In this plethora of existing tools, it is necessary for the humanoid robotics community to have a systematic evaluation to assist in choosing which of the available tools is best for their research. However, qualitative and quantitative evaluations are difficult to find.

To this end, Ivaldi et al. investigated the state of the art in dynamics simulation and reported a user-feedback based survey regarding the use of dynamics simulation tools in the robotics research community [19]. From the survey, it appears that the

major requirements for robotics researchers are better physics engines and open-source software.

Despite the numerous tools for simulation, the analysis of the user-based feedback survey in [19] indicated that there is not a general-purpose simulator which dominates the others in terms of performance or application. However, for humanoid robotics, Gazebo [20] emerges as the most popular among the open-source projects, while V-Rep and Webots [21, 22] are the preferred commercial simulators considering the user-based survey.

With a complementary point of view, this chapter is centered around a developer-based feedback survey to investigate the state-of-the-art in humanoid robotics simulation and software. It particularly aims to provide an outline concerning the special features and characteristics of current-day open source simulators with regards to the present research trends in humanoid robotics.

The chapter is organized as follows. Section 2 addresses the contemporary simulation scenarios and requirements of research on humanoid robotics. Section 3 lists the prominent open-source simulation packages, with a special emphasis on how they correspond to the current-day simulation scenarios. Physics engines are succinctly discussed in section 4 and structurally distinguished from built-in simulation packages. Finally, the chapter is concluded in section 5.

2 Contemporary Simulation Scenarios and Requirements

In order to portray the current state-of-the-art, the following questions were answered by the simulator developers. The questions were selected in a way so as to see how the simulators correspond to the current-day simulation requirements.

2.1 Actuator-Related Requirements

1. *Can the simulator incorporate actuator dynamics?*
   In an electrically-actuated robotic system, actuator dynamics represent the terms that account for rotor inertia, rotor friction, gear ratio, gear efficiency and electrical characteristics. In a hydraulically or pneumatically-actuated robotic system, more complicated actuator modeling may be required to provide a feasible and accurate simulation. Some simulators consider the actuator as a perfect torque source, while others employ exhaustive dynamics model that include actuator parameters.

2. *Can the simulator incorporate passive compliance?*
   The recent improvements in mechatronics technology allowed the creation of humanoid robots with physically compliant joints. Incorporation of physical compliance introduces numerous advantages [23]; however, the dynamics become

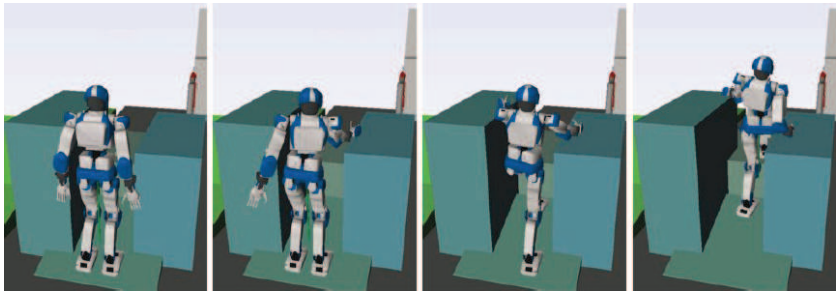

Fig. 2 Simulated HRP-4 humanoid is crumbling through an unstructured and narrow path [24]. Image courtesy of Karim Bouyarmane.

more complicated due the intrinsic passivity. The characterization of passivity within the dynamics framework poses additional challenges to the simulation.

3. *Is actual torque control available?*
   In some tasks, the robot is required to interact with the environment and exert a certain amount of force while there may be no specific position reference available. To this end, torque control mode is crucial for realizing the task. Simulator packages that surpass the torque control may solely provide position control option, and thus, cannot be utilized for such tasks.

2.2 Environment-Related Requirements

4. *Can the simulator allow the creation of custom environments and terrains?*
   The ability to create custom terrains and environment is a fundamental feature that enables testing the performance of the robot controller in balancing, locomotion and whole-body movement scenarios. For instance, Fig. 2 displays four snapshots from a simulation study, in which the HRP-4 robot (from AIST, Japan) walks forward through a narrow and unstructured path [24].

5. *Can the simulator provide a set of different environments?*
   A set of built-in environment options is a useful asset for simulation packages. Options may vary from indoor environments, such as furnished houses and humanoid soccer pitches, to outdoor environments, e.g., muddy/rocky areas, sloped surfaces and template terrains.

6. *Can the simulator provide a set of tools and objects in the simulated environment?*

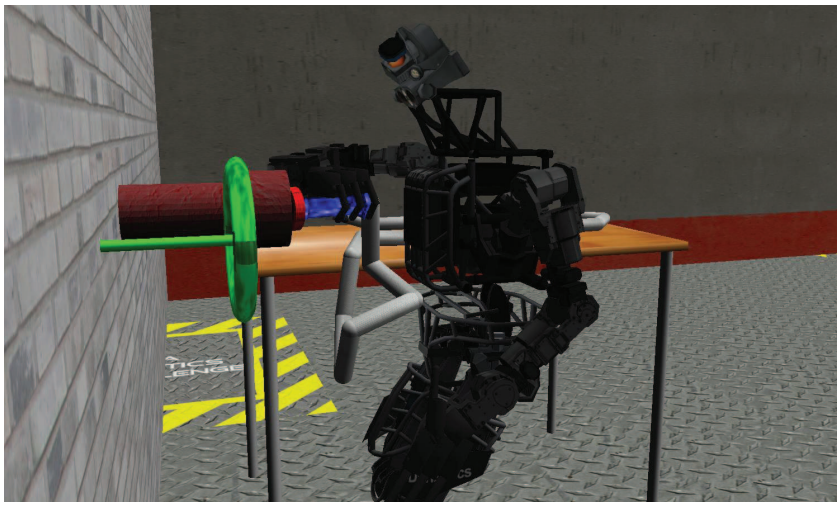

Fig. 3 Simulated Atlas humanoid is ejecting a hose from its socket. Image courtesy of DARPA. (http://www.DARPA.mil/)

With the recent DARPA Challenge[1], disaster response applications gained importance. Therefore, humanoid robots are expected to accomplish challenging tasks that involve the active use of various tools and objects in the simulated environment, e.g., driving a car, turning a valve, opening/closing doors, wielding a hose. Fig. 3 displays a simulation scenario in which the humanoid Atlas (from Boston Dynamics, US) is ejecting a hose from its socket.

7. *Can the simulator generate perturbations on the robot?*
   For robustness tests, one may need to perturb the simulated humanoid robot, for example with an external force applied to an arbitrary point on the robot. The ability to generate custom perturbations and add noise is certainly advantageous to test the robustness of the robot control.

2.3 Modeling-Related Requirements

8. *Can the simulator incorporate human model?*
   The latest developments in human-friendly robotics multiplied the possibilities of safe and sound interactions between humanoids and humans. Physical human-humanoid interaction and cooperation is an active research field that is more and more requiring simulators that can incorporate human models. Human models

[1] http://www.theroboticschallenge.org/

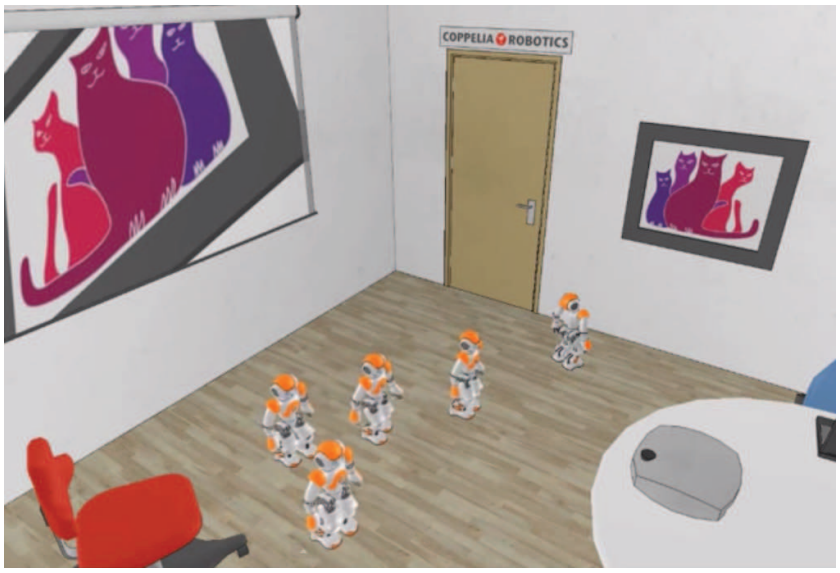

Fig. 4 Multiple NAO humanoids are walking in an office environment in V-Rep. In this scenario, one of the robots' vision is reflected to the projector. Image courtesy of Marc Freese, Coppelia Robotics.

can be also useful for ergonomy studies, for example to simulate human efforts in whole-body movements.

9. *Can the simulator simulate multiple humanoids?*
   The first simulators were usually centered around a single robot model, therefore they were unfit for studying scenarios in which multiple humanoids need to cooperate to accomplish a given task, e.g., soccer-playing humanoids. To address the multi-robot scenarios, most of the recent simulators have the capability of simultaneously simulating multiple robots. An example is displayed in 4, where five NAO humanoids are walking in an office environment.

10. *Can the simulator incorporate actual humanoid geometry?*
    The actual robot geometry may not be an essential factor in dynamic simulation of a humanoid, as long as dynamic and kinematic parameters are accurately included. However, one on one characterization of robot-specific humanoid geometry may be required when considering collisions with the environment. If we give an example, Fig. 3 shows a simulation in which the simulator includes the exact robot geometry. On the contrary, Fig. 5 displays a biped robot with its simulation model in which the actual geometry could not be characterized.

11. *Can the simulator enable imports from CAD software?*
    In order to obtain a fast and accurate simulation model, recent simulation pack-

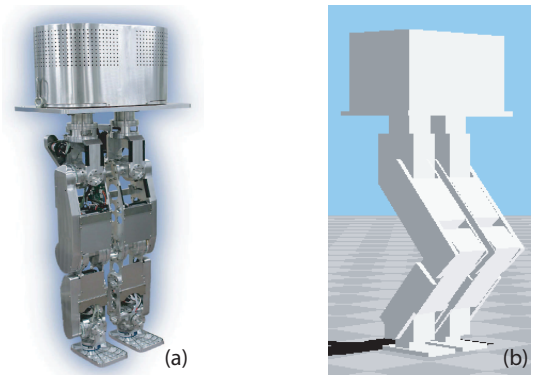

Fig. 5 a) The bipedal robot MARI-3, b) MARI-3's ROCOS model [25]. The actual robot geometry is not characterized; however, the simulation model is able to represent locomotion behavior of MARI-3.

ages allow direct imports from CAD software. This feature not only shortens the exhaustive model building process, but also enables the direct insertion of robot-specific properties, such as link parameters, robot geometry, and joint structure.

12. *Does the simulator provide common humanoid models?*
In recent years, the use of common hardware platforms gained importance in the field. Atlas, Hubo, Darwin-OP, Nao and HRP-4 maybe listed in this category since they are used as test platforms in several laboratories across the world. With this in mind, most simulators are already equipped with the built-in models of common humanoids. For instance, Fig. 3 and Fig. 6 display the simulated Atlas and Hubo robots in Gazebo and K'lampt environments, respectively.

2.4 Interfacing and Computation-Related Requirements

13. *Can the simulator run multiple simulations sequentially?*
In machine learning-based robot control, multiple experiments should be sequentially conducted. It is favorable for a simulator to automate this process, so as to prevent exhaustive and time-consuming manual tasks.

14. *Does the simulator have a MATLAB/Simulink Interface?*
MATLAB/Simulink provides a flexible and convenient environment in synthesizing controllers, due to its extended library, graphical editor and efficient solvers. Therefore, integration of a MATLAB/Simulink interface enables users to incor-

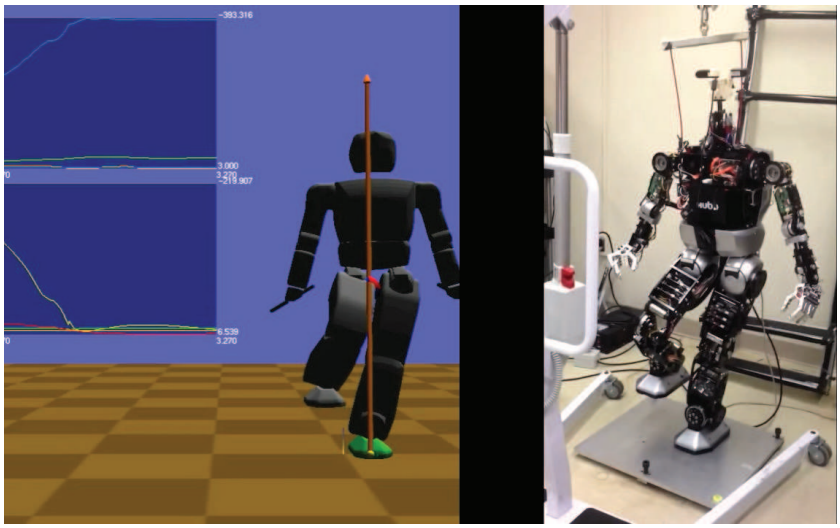

Fig. 6 The humanoid robot Hubo is one of the template models in K'lampt [26]. In this study, both the simulated robot and actual robot simultaneously perform balancing on a single leg. The accuracy of the simulator is verified by comparing the actual sensory data and simulated sensory data. Image courtesy of Kris Hauser.

porate these advantages for humanoid robotic simulations.

15. *Does the simulator have a ROS Interface?*
    ROS (Robot Operating System) is the most common open-source meta-operating system for robotics, providing numerous tools and libraries for robotic applications. Due to this fact, simulators often provide a ROS interface to benefit the computational advantages it presents.

3 Open Source Humanoid Robot Simulators

There is a great diversity of open source robot simulators. In [19], Ivaldi et al. report more than 40 active software projects. This section will provide an overview of the most important **open-source** simulators that are relevant for **humanoid robotics**. Commercial simulators that allows certain groups (academics, students) to use complete simulation features are also included. The following list follows no specific order.

ROCOS (Robot Control Simulator) is a dynamic simulator, developed by Y. Fujimoto and A. Kawamura[2]. It includes models of friction and collision between the

[2] http://www.kawalab.dnj.ynu.ac.jp

robot legs and the floor, and it is mainly used for research in multi-modal locomotion, e.g., running, jumping and walking [18].

OpenHRP3 (Open Architecture Human-centered Robotics Platform version 3) is an integrated software platform for robot simulations. It was developed in Advanced Institute of Science and Technology (AIST), Japan[3]. It consists of several modules, to name a few, a dynamics simulator, a control interface with the robot, a collision detector and several GUI interfaces. It can be used for a seamless simulation and control of the robot [27–29].

Robotran is a software that generates symbolic models of multi-body systems, which can be analyzed and simulated in Matlab and Simulink[4]. It is developed by the Center for Research in Mechatronics, Université Catholique de Louvain [30].

Gazebo is a multi-robot simulator for outdoor environments, developed by the Open-Source Robotics Foundation [20][5]. It is the official software tool for the DARPA Robotics Challenge [31]. It supports multiple physics engines (ODE, Bullet, DART) and, thanks to its modular and plugin-based structure, can be easily extended with new features. For example, a special plugin (`gazebo_yarp_plugin`) has been developed for the simulation of humanoid robots based on YARP, namely iCub and COMAN [32]. In particular, this development is gradually replacing the previous iCub simulator, **iCub_SIM**, that is based on ODE and is used for prototyping behaviors thanks to its seamless interface (the same code runs on the simulator and on the real robot)[6].

Morse is a generic simulator for academic research, used to simulate small and large indoor and outdoor environments with multiple robots[7]. It provides interfaces for different mobile robots (not legged) such as the PR2 [33][8].

Klamp't (Kris' Locomotion and Manipulation Planning Toolbox) is an open source, cross-platform software package for modeling, simulating, and planning for complex robots, particularly for manipulation and locomotion tasks[9]. It has been used by members of Team DRC-Hubo in the DARPA Robotics Challenge. The simulator allows the robust rigid body simulation with triangle mesh / triangle mesh collisions. Furthermore, actual objects can be captured using a Kinect sensor and directly integrated to the simulator [26]. In Fig. 6, a simulated Hubo humanoid in Klamp't environment can be observed.

V-Rep is a simulator with multi-robot simulation capability, produced by Coppelia Robotics[10]. It provides an integrated development environment that is based on a distributed control architecture. Each object/model can be separately controlled via an embedded script, a plug-in, a ROS node, a remote API client, or a user-based

[3] http://www.openrtp.jp/openhrp3/jp/

[4] http://www.robotran.be/

[5] www.gazebosim.org

[6] http://eris.liralab.it/wiki/Simulator_README

[7] http://morse-simulator.github.io/

[8] https://github.com/morse-simulator/morse

[9] http://klampt.org

[10] http://www.coppeliarobotics.com

Fig. 7 The SL software is configured both as a real-time robot controller and as a simulator for the Hydraulic Quadruped HyQ. Image courtesy of Claudio Semini, Istituto Italiano di Tecnologia.

input. Similar to Gazebo, it supports multiple physics engines (ODE, Bullet, Vortex). It is not open-source but it has an academic license that makes it free for use in research and academia [21,34]. See Fig. 4, as an example from V-Rep environment, where multiple NAO humanoids are walking in an office environment.

SL is a simulation and control library developed mainly by S. Schaal and his research group. It was primarily developed as a real-time robot controller, and later augmented to be a physical simulator[11]. Therefore, one can compose a script to simulate robot behavior and then directly use the same code for real-time experimentation [35,36]. Fig. 7 displays a scene in which HyQ is being controlled via SL software.

OpenRave is an environment for simulating motion planning algorithms for robotics. It contains several models of industrial robots and targets robotics automation[12]. It provides many command line tools to simulate robots and motion planners. The run-time core is sufficiently compact to be used within controllers [37,38].

RBDL (Rigid Body Dynamics Library) is a dynamics library developed by M. Felis, containing efficient implementations of Featherstone's rigid body dynamics algorithms from his book [39][13]. It can load robot models from Lua scripts or URDF files.

[11] https://www-clmc.usc.edu/s̃schaal/pub/RSS-SL.key.pdf

[12] http://openrave.org/

[13] http://rbdl.bitbucket.org

Drake is not only a simulator but a toolbox for analyzing the dynamics of robots and building control systems for them[14]. It is maintained by R. Tedrake and his Robot Locomotion Group at the MIT Computer Science and Artificial Intelligence Lab. It has been used for the DARPA Robotics Challenge on their Atlas robot [40].

The **NTRT Simulator** (NASA Tensegrity Robotics toolkit) is a tensegrity-specific simulator built to run on top of the Bullet Physics Engine[15], version 2.82. It can simulate a wide range of multi-body systems, in particular, compliant systems [41].

SCS (Simulation Construction Set) is a Java library for creating simulators [16], constructed at the Florida Institute for Human and Machine Cognition (IHMC), US [42]. It makes use of the Featherstone's algorithms for rigid body dynamics.

OpenSim is a toolkit for musculoskeletal modeling and dynamic simulation of movement, developed at Stanford University and supported by the US National Institute of Health and by DARPA[17]. It is freely available, open-source, and extensible through user plug-ins. The physics engine of this project is **SimBody** [43].

Moby is a multi-rigid body dynamics simulator developed by the Positronics Lab at George Washington University[18]. It supports the simulation of deformable bodies, together with their interactions to rigid/articulated bodies) [44].

In order to provide a comprehensive review on these simulators, a survey that included the questions in section 2 was constructed. Inputs of the developers can be viewed in Tables 1-4. Particularly, Table 1 displays the simulators' actuator simulation capabilities. Table 2 shows the way they incorporate environment model and objects in the simulated environment. Table 3 depicts their human and humanoid modeling properties Table 4 reveals their interfacing features.

4 Open Source Physics Engines

Fig. 8 presents a descriptive classification of the dynamics simulation tools. We can distinguish between *physics engines* (e.g. ODE, Bullet) and more complex software, that we call here "*system simulators*" (e.g. Gazebo, V-Rep, iCub_SIM) that are based on a physics engine.

System simulators make use of physics engines to simulate the dynamics of the bodies in the environment, but also provide additional features, such as sensor simulation, model editors, GUI interfaces and interaction with the operator. In some cases they also provide specific interfaces that facilitate seamless simulation and control of the robot and its virtual character. For example, they emulate the driver interfaces of the real robot in such a way that code running for the simulators can be

[14] http://drake.mit.edu

[15] https://github.com/NASA-Tensegrity-Robotics-Toolkit

[16] https://ihmcrobotics.github.io

[17] http://opensim.stanford.edu

[18] https://github.com/PositronicsLab/Moby

Tool	Incorporates actuator dynamics	Simulates passive compliance	Has torque control
ROCOS		×	×
OpenHRP3			×
Robotran	×	×	×
Gazebo			×
Morse			×
Klamp't	×	×	×
V-Rep	×	×	×
Gazebo + YARP			×
SL	×	×	×
OpenRAVE	×		
RBDL	×	×	×
Drake	×	×	×
iCub_SIM		×	×
NASA Tens.RT	×	×	×
SCS	×	×	×
OpenSim	×	×	×
Moby	×	×	×

Table 1 Actuation simulation capabilities.

Tool	Can create new environments and terrains	Provides a set of different environments	Provides tools in the environment	Can generate perturbations on the robot
ROCOS				×
OpenHRP3		×		×
Robotran				×
Gazebo		×	×	×
Morse	×	×	×	×
Klamp't	×			×
V-Rep	×	×	×	×
Gazebo + YARP	×	×	×	×
SL	×	×	×	×
OpenRAVE	×	×		
RBDL	×			×
Drake	×	×	×	×
iCub_SIM	×			
NASA Tens. RT	×			×
SCS	×			×
OpenSim	×	×	×	×
Moby	×			×

Table 2 Simulation of the environment.

Tool	Incorporates human model	Simulate multiple humanoids	Includes robot geometry	Can import from CAD	Includes common humanoid models
ROCOS					
OpenHRP3		×			
Robotran		×			
Gazebo			×		×
Morse	×	×	×	×	
Klamp't		×	×	×	×
V-Rep	×	×	×		×
Gazebo + YARP		×	×	×	×
SL	×	×	×		×
OpenRAVE		×	×		
RBDL	×				
Drake		×	×	×	×
iCub_SIM	×		×		
NASA Tens. RT		×			
SCS	×	×	×		×
OpenSim	×		×	×	
Moby	×	×	×		×

Table 3 Human and humanoid models.

Tool	Can run multiple simulations in sequence	Has a Matlab/Simulink interface	Has a ROS interface
ROCOS			
OpenHRP3	×		
Robotran	×	×	
Gazebo			×
Morse	×		×
Klamp't	×		×
V-Rep	×	×	×
Gazebo + YARP		×	×
SL	×	×	×
OpenRAVE	×	×	×
RBDL	×		×
Drake	×	×	×
iCub_SIM		×	
NASA Tens. RT	×		
SCS	×		
OpenSim	×	×	
Moby	×		

Table 4 Simulation interfaces.

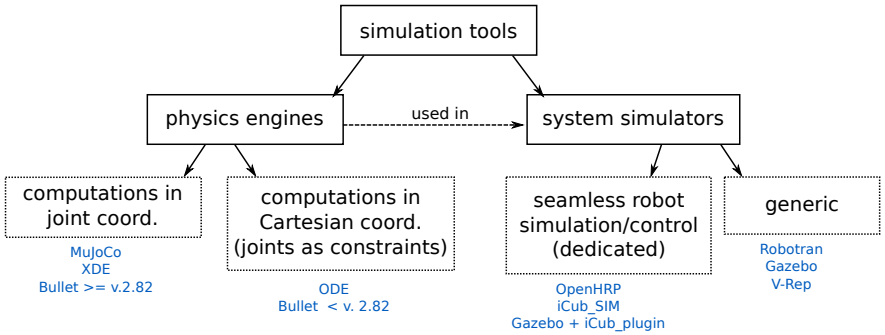

Fig. 8 From [19]. A practical classification of the simulation tools.

switched to the robot at no cost. This makes them in practice the "official" tools for some platforms; for example, OpenHRP for the HRP robot series [29], SL simulator for Sarcos humanoid [35], iCub_SIM [45] and the more recent Gazebo plugin for iCub [32].

Physics engines can be discriminated by the way they represent rigid-body structures, see Fig. 8. On one hand, we have software tools like ODE which represents joints as constraints between bodies; on the other hand, we have software like XDE, OpenHRP, which make use of parametrized rigid-body dynamics representations, where joints are simply part of the robotics structure.

The second group benefits from the straightforward computation of quantities that recur in robot control, such as Jacobian and joint-space mass matrices and other physical quantities.

The critical difference between the two classes is in the way contact forces are computed. The first class considers contacts forces as bilateral/unilateral constraints, which are added to the list of constraints used to describe the joints; then the same solver is used to find the forces for the global system, including contacts and joints. In the second class, on the contrary, only constraints from the contacts are solved, which notably simplifies the problem. In general, finding the correct contact forces can be burdensome. Contact modeling and simulation is an area of research on its own, which is actively explored improving contact models [26,46] and solvers [47].

The most prominent physics engines may be listed as ODE, PhysX, Bullet, DART.

ODE[19] (Open Dynamics Engine) is an open-source library for simulating rigid body dynamics, with a built-in collision detector. It is one of the most known multi-purpose rigid-body physics engines, used in many computer games and embedded in several simulation tools.

PhysX[20] is an engine developed by NVIDIA for enabling real-time physics in video-games. It is optimized for GPU (Graphics Processing Unit) computations.

[19] http://www.ode.org/

[20] https://developer.nvidia.com/physx-sdk

Bullet[21] is another open-source physics library, mostly used for computer graphics and animation. As ODE and NVidia PhysX, it was a game-oriented engine, that enforced joints constraints numerically. The latest release (v.2.82) also supports Featherstone's Articulated Body Algorithm [48] and a Mixed Linear Complementarity Problem. These make it more suitable for robotics applications, since dynamics is solved in joint coordinates and contacts resolution is more stable.

DART[22] (Dynamic Animation and Robotics Toolkit) is another open-source physics engine, used for robotics and computer graphics. For accuracy, it utilizes Lagrange's equations derived from D'Alembert's principle to describe the dynamics of motion.

SimBody[23] is an open-source C++ API that implements Featherstone's algorithms for rigid body mechanics, with the support of different contact models. The supported contact models are one of the interesting features of SimBody; one can model contacts through Hertz contact theory, the elastic foundation model, and can consider Stribeck friction and Hunt and Crossley dissipation models [49].

A quantitative comparison among all simulators is very difficult to obtain, as not all simulators may necessarily have the same features of parameters. However, an interesting comparison among physics engine was recently presented by Erez et al. [50], where the authors compared the performances of **MuJoCo**[24] with Bullet, Havok, ODE and PhysX. According to their study, MuJoCo, which is optimized for robot simulation, outperforms the other engines. We report, for the interested reader, the results of their comparisons on a simulated humanoid robot in Fig. 9: from left to right, the first image is the simulated humanoid in contact with the floor, the second image is the plot of the raw speed as thousands of evaluations per second for each engine, while the third image is the plot of speed-accuracy trade-offs. More details can be found in [50].

5 Concluding Remarks

In this chapter, we provide an outline for the current state of the art in open source humanoid simulation software libraries, with a special emphasis on the simulation requirements for the main ongoing researches. The discussion is centered around a survey that was answered by the developers, where we investigated how the simulators address the needs of recent research trends in the field. Based on the survey inputs, we summarized the listed simulators' special features and capabilities. In addition, a succinct review was addressed regarding the open source physics engines that are at the basis of the current simulators and may be used to build new simulation software.

[21] http://bulletphysics.org/wordpress/

[22] http://dartsim.github.io/

[23] https://simtk.org/home/simbody/

[24] http://www.mujoco.org/

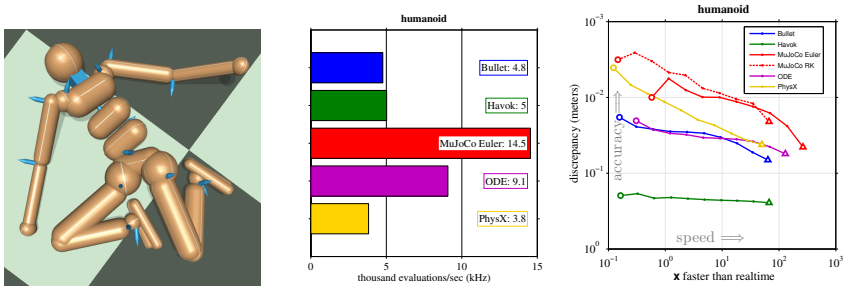

Fig. 9 Image courtesy of Tom Erez, Yuval Tassa and Emanuel Todorov [50]. The first image illustrates the simulated humanoid. The second image shows raw speed as thousands of evaluations per second for each engine. The third image shows the speed-accuracy trade-off in terms of our consistency measure.

This chapter does not discuss how close to reality is a simulator. Such an assessment should require a detailed and exhaustive study using the real humanoid robots and their simulated models, to assess the so called *reality gap*. Comparing the actual and simulated sensory output, it is possible to assess the reality gap for a given simulator. For instance, researchers performed simultaneous one-legged balancing experiments on a real Hubo humanoid and its Klamp't model; see Fig. 6. Although the aforementioned study may be enriched with the addition of more dynamic motion patterns (walking, jumping, pushing) for further investigation, it could be taken as a basis for other simulator assessment studies.

The field of simulation software development is quite active and vibrant. More often than not, a new simulator software is presented to the community or additional improvements are announced on the existing ones. Keeping this in mind, this chapter is written to capture the current-day simulation requirements, with a glimpse on the possible future trends in the field.

Indeed, current simulators are quite advanced; however there is room for further improvement in simulating complex systems. For instance, soft actuators, deformable materials and generally soft-bodied robots are still difficult to simulate [51]. The reality gap for soft systems and actuators being larger, it could be interesting to compensate the inaccurate simulation with better control and machine learning techniques; e.g., designing controllers in simulation that are robust to differences to the real robot [52], or learning the difference between simulation and reality [53], that was recently proven effective also for damaged robots.

Another point to improve for simulators is to increase computational efficiency, such that they can be used in real-time for remote control applications or in model-predictive control schemes. This a crucial feature that is particularly important for humanoid robots performing complex whole-body dynamic movements [54].

Currently, it is difficult to single out a certain simulator software that is capable of addressing all the requirements. Despite this fact, we strongly believe that this chapter provides guidance to the interested researcher in choosing the right simulator package that suits to his/her specific research objectives.

References

1. B. Lint and T. Agerwala. Communication issues in the design and analysis of parallel algorithms. *IEEE Trans. on Software Engineering*, SE-7(2):174–188, 1981.
2. C. S. G. Lee and P. R. Chang. Efficient parallel algorithms for robot forward dynamics computation. *IEEE Trans. on Systems, Man, and Cybernetics*, 18(2):238–251, 1988.
3. A. Fijany and A. Bejczy. A class of parallel algorithms for computation of the manipulator inertia matrix. *IEEE Trans. on Robotics and Automation*, 5(2):600–615, 1989.
4. S. McMillan, D. E. Orin, and P. Sadayappan. Toward super-real-time simulation of robotic mechanisms using a parallel integration method. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(2):384–391, 1992.
5. A. Fijany, I. Sharf, and G. M. T. D'Eleuterio. Parallel o (log n) algorithms for computation of manipulator forward dynamics. *IEEE Trans. on Robotics and Automation*, 11(3):389–400, 1989.
6. A. Aghili. A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation. *IEEE Trans. on Robotics*, 21(5):834–849, 2005.
7. K. Yamane and Y. Nakamura. Comparative study on serial and parallel forward dynamics algorithms for kinematic chains. *Int. Journal of Robotics Research*, 28(5):622–629, 2009.
8. J. Y. S. Luh, M. W. Walker, and R. P. Paul. On-line computation scheme for mechanical manipulator. *Journal of Dynamic Systems, Measurement, and Control ASME Trans.*, 102(2):69–76, 1988.
9. R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *Int. Journal of Robotics Research*, 2(1):13–30, 1983.
10. O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Trans. on Robotics and Automation*, 3(1):43–53, 1988.
11. G. Rodriguez. Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics. *IEEE Trans. on Robotics and Automation*, 3(6):624–639, 1988.
12. R. Featherstone. A beginner's guide to 6-d vectors (part 1). *IEEE Robotics and Automation Magazine*, 17(3):83–94, 2010.
13. M. W. Walker and D. E. Orin. Efficient dynamic computer simulation of robotics mechanisms. *Journal of Dynamic Systems, Measurement, and Control ASME Trans.*, 104(2):205–211, 1988.
14. Y. Fujimoto and A. Kawamura. Robust biped walking with force interaction control between foot and ground. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 2030–2035, Leuven, Belgium, May 1998.
15. J. Nakanishi, M. Mistry, and S. Schaal. Inverse dynamics control with floating base and constraints. In *Proc. IEEE Intl. Conf. on Robotics and Automation(ICRA)*, pages 1942–1947, Rome, Italy, May 2007.
16. M. Mistry, J. Buchli, and S. Schaal. Inverse dynamics control of floating base systems using orthogonal decomposition. In *Proc. IEEE Intl. Conf. on Robotics and Automation(ICRA)*, pages 3406–3412, Anchorage, US, May 2010.
17. Y. Nakamura, H. Hirukawa, K. Yamane, S. Kajta, K. Fujiwara, F. Kanehiro, F. Nagashima, Y. Murase, and M. Inaba. Humanoid robot simulator for the meti hrp project. *Robotics and Autonomous Systems*, 37(2-3):101–114, 2001.
18. Y. Fujimoto and A. Kawamura. Simulation of an autonomous biped walking robot including environmental force interaction. *IEEE Robotics and Automation Magazine*, 5(2):33–42, 1998.
19. S. Ivaldi, J. Peters, V. Padois, and F. Nori. Tools for simulating humanoid robot dynamics: A survey based on user feedback. In *Proc. IEEE Intl. Conf. on Humanoid Robotics*, pages 842–849, Madrid, Spain, December 2014.
20. N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 2149–2154, Sendai, Japan, September 2004.
21. E. Rohmer, S. P. N. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1321–1326, Tokyo, Japan, November 2013.

22. O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.
23. B. Ugurlu, J. A. Saglia, N. G. Tsagarakis, S. Morfey, and D. G. Caldwell. Bipedal hopping pattern generation for passively compliant humanoids: Exploiting the resonance. *IEEE Transactions on Industrial Electronics*, 61(10):5431–5443, 2014.
24. K. Bouyarmane, J. Vaillant, F. Keith, and A. Kheddar. Exploring humanoid robot locomotion capabilities in virtual disaster response scenarios. In *Proc. IEEE Intl. Conf. on Humanoid Robotics*, pages 337–342, Osaka, Japan, December 2012.
25. B. Ugurlu and A. Kawamura. A unified control frame for stable bipedal walking. In *Proc. IEEE Intl. Conf. on Industrial Electronics and Control(IECON)*, pages 4167–4172, Porto, Portugal, November 2009.
26. K. Hauser. Fast interpolation and time-optimization with contact. *Int. Journal of Robotics Research*, 33(9):1231–1250, 2014.
27. F. Kanehiro, H. Hirukawa, and S. Kajita. Openhrp: Open architecture humanoid robotics platform. *Int. Journal of Robotics Research*, 23(2):155–165, 2004.
28. K. Yamane and Y. Nakamura. Parallel o(logn) algorithm for dynamics simulation of humanoid robots. In *Proc. IEEE Intl Conf. on Humanoid Robotics*, pages 554–559, Genoa, Italy, December 2006.
29. S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita, and H. Hirukawa. Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 3641–3647, San Diego, CA, US, October 2007.
30. P. Fisette and J. C. Samin. Robotran: Symbolic generation of multi-body system dynamic equations. *Advanced Multibody System Dynamics*, 20:373–378, 1993.
31. C. E. Aguero, N. Koenig, H. Boyer I. Chen, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J. L. Rivero, J. Manzo, E. Krotkov, and G. Pratt. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *IEEE Trans. on Automation Science and Engineering*, 12(2):494–506, 2015.
32. E. Mingo Hoffman, S. Traversaro, A. Rocchi, M. Ferrati, A. Settimi, F. Romano, L. Natale, A. Bicchi, F. Nori, and N. Tsagarakis. Yarp based plugins for gazebo simulator. In *Modelling and Simulation for Autonomous Systems Workshop (MESAS)*, 2014.
33. G. Echeverria, S. Lemaignan, A. Degroote, S. Lacroix, M. Karg, P. Koch, C. Lesire, and S. Stinckwich. Simulating complex robotic scenarios with morse. In *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 197–208, 2012.
34. M. Freese, S. Singh, A. Degroote, S. Ozaki, and N. Matsuhira. Virtual robot experimentation platform v-rep: A versatile 3d robot simulator. In *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 51–62, 2010.
35. S. Schaal. *The SL simulation and real-time control software package*. Technical report, Univ. of Southern California, California, US, 2001.
36. A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47(2-3):69–77, 2004.
37. R. Diankov and J. Kuffner. *OpenRAVE: A Planning Architecture for Autonomous Robotics*. Technical report, Robotics Institute, Carnegie Mellon University, Pittsburgh, US, 2008.
38. Q.-C Pham and Y. Nakamura. Time-optimal path parameterization for critically dynamic motions of humanoid robots. In *Proc. IEEE Intl. Conf. on Humanoid Robotics*, pages 165–170, Osaka, Japan, November 2012.
39. Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
40. R. Tedrake. *Drake - A planning, control, and analysis toolbox for nonlinear dynamical systems*. Technical report, Massachusetts Institute of Technology, Massachusetts, US, 2014.
41. K. Caluwaerts, J. Despraz, A. Iscen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral. Design and control of compliant tensegrity robots through simulation and hardware validation. *Journal of the Royal Society Interface*, 11(98):1742–1757, 2014.

42. T. Koolen, S. Bertrand, G. Thomas, T. Wu, J. Smith, J. Englsberger, and J. Pratt. Design of a momentum-based control framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics*, 13(1):1650007–34, 2016.
43. S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Trans. on Biomedical Engineering*, 54(11):1940–1950, 2007.
44. E. Drumwright. A fast and stable penalty method for rigid body simulation. *IEEE Trans. on Visualization and Computer Graphics*, 14(1):231–240, 2008.
45. V. Tikhanoff, A. Cangelosi, and G. Metta. Integration of speech and action in humanoid robots: icub simulation experiments. *IEEE Trans. on Autonomous Mental Development*, 3(1):17–29, 2010.
46. E. Drumwright and D.A. Shell. Extensive analysis of linear complementarity problem (lcp) solver performance on randomly generated rigid body contact problems. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5034–5039, 2012.
47. E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5026–5033, Algarve, Portugal, October 2012.
48. R. Featherstone and D. E. Orin. *Handbook of Robotics*, chapter Dynamics, pages 35–65. B. Siciliano and O. Khatib Eds.,, 2008.
49. C. Canudas de Wit, H. Olsson, K. J. Astrom, and P. Lischinsky. A new model for control of systems with friction. *IEEE Trans. on Automatic Control*, 40(3):419–425, 1995.
50. T. Erez, Y. Tassa, and E. Todorov. Simulation tools for model-based robotics: comparison of bullet, havok, mujoco, ode and physx. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, May 2015.
51. Frederick Largilliere, Valerian Verona, Eulalie Coevoet, Mario Sanz-Lopez, Jeremie Dequidt, and Christian Duriez. Real-time Control of Soft-Robots using Asynchronous Finite Element Modeling. In *ICRA 2015*, page 6, SEATTLE, United States, 2015.
52. Andrea Del Prete and Nicolas Mansard. Robustness to Joint-Torque Tracking Errors in Task-Space Inverse Dynamics. *IEEE Transaction on Robotics*, 2016.
53. A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521:503–507, 2015.
54. Serena Ivaldi, Jan Babič, Michael Mistry, and Robin Murphy. Special issue on whole-body control of contacts and dynamics for humanoid robots. *Autonomous Robots*, 40(3):425–428, 2016.